CODE REVIEW REPORT V2

CORSAIR-Support

The Code Review Process

Code review is a specialized task with the goal of identifying types of weaknesses that exist within a given code base. A Secure Code Review does not attempt to identify every issue in the code but instead attempts to identify types of risk within the code such that mitigation strategies can be devised.

During the actual review, members of a review team review the application code for security problems and categorize the findings based on the weakness categories (e.g., authentication, authorization, etc.).

It should be noted that while the review process will be as thorough as possible in finding and reporting security weaknesses, it is not guaranteed to always find every possible weakness.

The actual review involved a manual investigation of the Laravel code of the *development* branch of git Repo https://github.com/corsairnow/Corsair-Support

Reviewer Detail

This code has been reviewed by Ms Ruchita Sheth. Experienced and Professional Developer closed to 7yrs of experience. Her expertise is insecure coding development and reviewing source code as well as developing applications for various sectors.

Stackoverflow Profile

V1

The version one of the report has already been submitted and this report is being prepared to submit the review of below mentioned points

1. Overall project structure
2. Database Structure
3. Data security
4. Usage of
    a. CSS
    b. AJAX
    c. JS
    d. XML
    e. JSON
    f. SQL
5. APIS used internally and Externally
6. UI UX design reviews

As mentioned in the previous report, Code seems to look good. Neither the best nor the poor quality. There are scope of improvements that can scale up the project's structure and understanding of code in an easy way to any new developer.

Database Structure

In the provided application MySql has been used to store data. As the application is implemented in Laravel tables for the database are created using migration files and as the code shows all the tables are created properly using migration files.

- Table structure looks good
- As naming convention columns name are using snake case
- Each table have columns to record time of creating and / or updating of a data

But,

- In the `routes` table all the columns are allowed to be set as null other than primary key. This should not be the case.
- Foreign keys in some of the tables are missing. This will make some issue when the data is deleted from the parent table and not from the child table neither by cascade nor by programming
- It is advisable to apply indexing on the tables to reduce the timing to fetch data from the tables when records are in huge numbers.

Database Security

Security of data is a top priority nowadays. Sometimes it is enforced by external rules and regulation or sometimes because the application owner cares about the data and privacy of the customers

To make sure data in a database are sequel check the below checklist:

1. Remove users of a database those can access a database without a password
2. Remove test database that any user have access to
3. Make database access available only through specific host
4. Store all data encrypted

Data in user's tables like email can be encrypted so in case if data is stolen neither of a hacker can have a customer's emails without encryption key.

There isn't any possibility of SQL injection as there is no direct SQL query written. All the input data are binded properly into SQL queries.

CSS

CSS stands for Cascade Style Sheet that is being used to decorate a web page with different style written in.

There are some key points that need to be taken care of while writing a CSS.

1. Code should not contain inline style sheet
2. Code should not contain deprecated elements & attributes.
3. Code should be written in easy and understandable way.
4. Developer should be use shorthands to reduce the lines of rules. Some of the CSS properties like paddings, margins, fonts and borders can be written in a much simpler way by shorthands.
5. Add comments when necessary because in some cases, we may need to write additional explanations.

After reviewing the code there are some pages found that use an inline CSS.

Source File	Line Number
resources\views\backend\super_user\settings\2fa.blade.php	162, 167, 234, 284, 286, 287
resources\views\frontend\support_manager\domains\edit.blade.php	133
resources\views\frontend\support_manager\domains\create.blade.php	129
resources\views\frontend\support_agent\tickets\bulk-tickets.blade.php	339
resources\views\frontend\platform\ticket.blade.php	62
resources\views\frontend\member\ticket\index.blade.php	44, 45
resources\views\frontend\member\ticket\show.blade.php	100
resources\views\git_book\single_page.blade.php	15, 25, 28, 43, 53
resources\views\frontend\content_manager\category\edit_category_form.blade.php	56
resources\views\frontend\content_manager\category\create_category_form.blade.php	8, 53

In this project, 90% of CSS was applied at external and rest of 10% was applied at inline. As some pages where add inline CSS with their line number are mentioned above of the table

Developers should be try to reduce the insert inline CSS.

JavaScript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions.

There are some key points that need to be taken care of while writing a JavaScript.

1. Variable should be initialize and declare at the top
2. Remove the duplicate code. Developer should move the duplicated code into a function and call the function in all the instances that it was used before.
3. Add comments when necessary because in some cases, we may need to write additional explanations.
4. After applying the code, check code on the console for errors.
5. Try to write understandable code that other developers find code easily.

After reviewing the code there are some pages found that use an internal JavaScript which are used as external JavaScript.

Findings

Source File	Line Number
resources/views/auth/login.blade.php	58
resources/views/backend/super_user/contents/create.blade.php	110, 113, 114
resources/views/backend/super_user/contents/edit.blade.php	114, 117, 118
resources/views/backend/super_user/contents/editor.blade.php	108 to 111
resources/views/backend/super_user/contents/layout.blade.php	41
resources/views/frontend/member/csr_token_holder/index.blade.php	191 to 196, 204
resources/views/frontend/member/ticket/index.blade.php	97 to 106
resources/views/frontend/member/dashboard.blade.php	201 to 210
resources/views/frontend/member/home.blade.php	51
resources/views/frontend/member/login.blade.php	58
resources/views/frontend/support_agent/csr_token_holder/import.blade.php	142 to 144, 146
resources/views/frontend/support_agent/csr_token_holder/index.blade.php	64 to 69
resources/views/frontend/support_agent/csr_token_holder/settings.blade.php	Empty script tag found

resources/views/frontend/support_agent/templates/logs.blade.php	117 to 126
resources/views/frontend/support_agent/tickets/index.blade.php	181 to 184, 188, 189, 192 to 195
resources/views/frontend/support_agent/dashboard.blade.php	137 to 146
resources/views/frontend/support_manager/csr_token_holder/settings.blade.php	92
resources/views/frontend/support_manager/settings.blade.php	105, 106
resources/views/frontend/support_manager/tickets.blade.php	126 to 135
resources/views/login_request.blade.php	35

AJAX

AJAX is used to make operations in the backend like selecting, updating or deleting data without refreshing a web page and displaying changes related to them in a Frontend design.

Reviewing a code for an ajax it looks batter user of AJAX at all places.

There are some ajax call those needs improvements in the re-use of code. The same AJAX setup has been found written in multiple places.

Source File	Line Number
resources/views/backend/super_user/menu/index.blade.php	328
resources/views/layouts/frontend/master.blade.php	163
public/frontend/assets/js/common.js	5
resources/views/backend/super_user/users/index.blade.php	96

```
$.ajaxSetup({
   headers: {
       'X-CSRF-TOKEN': "{{csrf_token()}}",
   }
});
```

The above code block found multiple time written in js and a view files. Instead move it to a common js file which gets loaded on each page of the web application

SQL

SQL is a programming language used by nearly all relational databases to query, manipulate, and define data, and to provide access control.

1. Programmers needs to write optimized SQL queries for integration with other applications
2. Programmers needs to maintaining data quality and overseeing database security.
3. Use simple and easily understood names for columns and tables.
4. Avoid organizing your database into tables that contain many one- or two-column tables. For example, data such as dates and zip codes don't need their own tables with foreign keys.

I tested all the above points in our web app and get the following result:

Description	Result
SQL query checking according to Laravel SQL query structure	Passed

HTML

HTML (Hyper Text Markup Language) is used to create the structure of the WebPages. Almost every web page, user interacts with, will be in HTML.

There are some key points that need to be taken care of while writing a HTML:

1. Code should be simple and easy to understand.
2. Always close the HTML tags. For ex: if you start <a> tag you need to place the corresponding closing tag at the end.
3. Use comments as much as required for make your HTML code clearer for others.
4. Use width and height for images.
5. Use name of classes and ids with their corresponding section for easy to understand.

I tested all the above points in our web app and get the following result:

Description	Result
Checked resources\views\frontend\support_manager\dashboard.blade.php file	Passed
Checked resources\views\frontend\support_agent\dashboard.blade.php file	Passed
Checked resources\views\frontend\theme.blade.php file	Passed

Internal APIs

An internal API is an interface that enables access to a company's backend information and application functionality for use by the organization's developers.

There are some key points that need to be taken care of while writing an Internal APIs:

1. Developers needs to fetch proper API on the database, without occurring the malicious attacks
2. APIs should be easily to discoverable and designed for the reusable.

We are not using any Internal API right now

External APIs

There are some key points that need to be taken care of while writing an external APIs

1. An external or open API is an API designed for access by a larger population as well as web developers.
2. Check the third-party external API keys for errors.

Encryption

Encryption protects the data from any third-party intervention. Through encryption, data becomes non-human readable and protects it while transferring or sending.

There are some key points that need to be taken care of while writing an encryption:

1. Developers should be secured all the data and tables that third party doesn't try to fetch all the information.
2. Developers need to encrypt the information end-to-end that it protects data while transferring and sending the data.
3. Server of every website should be encrypted to protect from the attackers.
4. Consider to use HTTPS, rather than HTTP.

I tested all the above points in our web app and get the following result:

Description	Result
SSL checking	Passed

Security - Backend

The backend is server-side. It's basically how the application works, applies the business logic, changes, and updates.

There are some key points that need to be taken care of while writing a security-backend:

1. Implement multi-factor authentication to prevent automated attacks.
2. Test the session timeout system and make sure the session token is invalidated after logout.

I tested all the above points in our web app and get the following result:

Description	Result
I checked google 2FA and its working perfectly	Passed
I tested session timeout and portal logged out successfully after given timeframe if idle	Passed

Security – Database

Database security will be to secure and maintain the integrity of the database. Any malicious input will be prevented through the security.

There are some key points that need to be taken care of while writing a security-database:

1. Programmers needs to get backup of all database in the case of corrupt files or missing
2. Databases require specialized security measures to keep them safe from cyber attacks.
3. Use the database and web applications firewalls.
4. Keep all the applications up-to date.

I tested all the above points in our web app and get the following result:

Description	Result
I tried to export database and its exported successfully without any risk took very less loading time.	Passed
Tried wrong login attempts and account get locked for a certain time frame due to security purposes.	Passed
database versions is up to date	Passed

Security – Frontend

Front-end web development is the development of the graphical user interface of a website, through the use of HTML, CSS, and JavaScript, so that users can view and interact with that website.

There are some key points that need to be taken care of while writing a security-frontend:

1. Sanitizing all inputs into your web application is a great way to prevent cross-site scripting attacks
2. Self-hosting your CSS files on your servers prevents you from falling victim to CSS injection-related attacks.
3. Use Captcha for public facing endpoints (login, registrations).
4. Use a modern framework that handles security automatically..
5. There should be a validation while uploading file. For ex: if there is option to upload image only no other file needs to upload on that page.
6. There should be validation of form fields JavaScript code, when programmer adds the code of JavaScript code on the form fields that code should not be show on the frontend.
7. User should be use only that emails which are exists in the real world of the portal.

I tested all the above points in our web app and get the following result:

Description	Result
All input fields are sanitizing properly before any backend operations	Passed
All CSS files are posted on our servers.	Passed
Captcha checking and its not implement	Failed
Tried uploading wrong extension file while creating ticket and its working fine	Passed
Validation checking	Passed
Tried none existed email and getting error	Passed